Modul Praktikum

Pemrograman Berorientasi Objek

Versi 1.0

Oleh:
Puji Hartono, ST

Materi:

1. Kelas dan Objek
2. Pewarisan
3. Polymorfisme
4. Pengkapsulan
5. Interface
6. Pemaketan

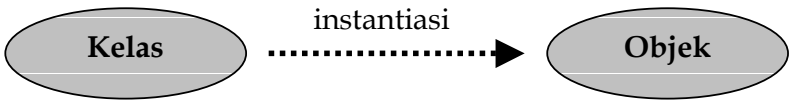
Modul 1. Kelas dan Objek

A. Tujuan

Praktikan diharapkan dapat:
1. Membuat kelas dengan beberapa variabel/atribut dan *metodh*
2. Menginstantiasi kelas menjadi objek

B. Teori Singkat

Dalam paradigma pemrograman berorientasi objek dikenal kelas dan objek. Kelas merupakan *blue print* dari objek-objek yang akan dibuat. Analogi kelas dan objek seperti *rancangan* model rumah dan *pembangunan* rumah-rumah, adapun proses pembuatan objek dari kelas dikenal dengan *instantiasi.*



Gambar 1. Pembuatan objek dari kelas dengan instantiasi

Sebagai contoh kita ambil kelas manusia. Kelas manusia mempunyai atribut : nama. Selain itu kelas manusia juga mempunyai metodh: tampilkanNama, kerja, makan. Kasus diatas diimplementasikan dalam bahasa Java sebagai berikut

```
/*
Disimpan dalam file "manusia.java"
*/

public class manusia
{
      public String nama;

      public manusia(String n)
      {
      this.nama = n;
      }

      public String tampilkanNama()
      {
      return nama;
      }


      public void makan()
      {
      System.out.println("Nyam... nyam... nyam...");
      }

      public void kerja()
      {
      System.out.println("Kerja...kerjaaa...");
      }

}
```

Adapun kode untuk menginstantiasi kelas manusia menjadi objek Andi yang mengimplementasikan metodh: tampilkanNama dan makan adalah sebagai berikut.

```
/*
Disimpan dalam file "andi.java"
*/

class andi
{
      public static void main(String arg[])
      {
      manusia andi= new manusia("Andi");
      System.out.println("Nama= "+ andi.tampilkanNama());
      andi.makan();
      }
}
```

Hasil eksekusi terhadap *class* andi adalah sebagai berikut:

```
Nama= Andi
Nyam... nyam... nyam...
```

Metodh Abstrak

Metodh abstract adalah metodh yang belum mempunyai implementasi. Pendefinisian metodh abstrak adalah dengan menambahkan keyword abstract, seperti contoh dibawah ini:

```
abstact void BerangkatKerja();
```

Pada metodh abstract ini tidak didefinisikan/implementasi metodh BerangkatKerja tersebut (misalkan apakah jalan kaki, naik angkot, naik motor, naik mobil atau dengan cara lainnya.

C. Praktikum

1. Buatlah kelas mahasiswa dengan atribut/variabel= nama dan npm dan mempunyai metodh: tampilkanNama, tampilkanNpm, belajar, olahRaga, makan, minum.
2. Buatlah 3 objek untuk menginstantiasi kelas mahasiswa dengan atribut nama dan npm Anda sendiri dan 2 teman Anda dan mengimplementasikan beberapa metodh yang telah Anda definisikan dalam kelas mahasiswa.

Kompile file kelas mahasiswa dan objek-objek yang telah Anda buat, kemudian jalankan objek-objek yang telah Anda buat !

Modul 2. Pewarisan

A. Tujuan

Praktikan diharapkan dapat:

3. Memahami dan mengimplementasikan konsep pewarisan
4. Memahami dan mengimplementasikan *this* dan *super* sesuai kebutuhan
5. Memahami dan mengimplementasikan *metodh overriding*

B. Teori Singkat

Salah satu kelebihan pemrograman berorientasi objek adalah penggunaan ulang kode-kode yang telah dibuat. Pewarisan adalah salah satu cara untuk menggunakan kode-kode yang telah dibuat sebelumnya.

Sebagai contoh kelas manusia diturunkan menjadi kelas: programmer, tentara.

```
/*
Disimpan dalam file "programmer.java"
*/

public class programmer extends manusia
{
      public programmer(String n)
      {
            super(n);
      }

      public void kerja()
      {
      System.out.println("Tak…Tak…Klik..");
      }

      public void bersantai()
      {
      System.out.println("Game over, You lose...");
      }
}
```

```
/*
Disimpan dalam file "tentara.java"
*/

class tentara extends manusia
{
      public String pangkat;
      public tentara(String n, String p)
      {
            super(n);
            this.pangkat= p;
      }



      public String tampilkanPangkat()
```

```
	{
	return pangkat;
	}

	public void kerja()
	{
	System.out.println("Dor... Dor... Dor..");
	}
}
```

Keyword super digunakan untuk memanggil *metodh* yang ada pada *baseclass, sedangkan this menunjukkan/mereferensi pada objek terkini*

Andi adalah seorang programmer keturunan manusia, setelah dia makan lalu kerja dan terakhir dia bersantai dengan komputernya sehingga implementasi dengan kode javanya.

```
/*
Disimpan dalam file "andi.java"
*/

class andi
{
	public static void main(String arg[])
	{
	programmer andi= new programmer("Andi");
	System.out.println("Nama= "+ andi.tampilkanNama());
	andi.makan();
	andi.kerja();
	andi.bersantai();
	}
}
```

Kalau dieksekusi *class* Andi akan menghasilkan

```
Nama= Andi
Nyam... nyam... nyam...
Tak.Tak..Klik..
Game over, You lose...
```

Setelah Objek andi memberitahu namanya, dia makan dengan *metodh* warisan dari kelas manusia, kemudian dia kerja dengan metodh khusus kelas programmer dan terakhir dia bersantai juga dengan metodh khusus kelas programmer.

Kelas turunan akan mewariskan atribut-atribut dan metodh-metodh *parentclass*nya/*baseclass*, akan tetapi dia tidak mewarisi konstruktor-konstruktornya sehingga ketika andi makan maka dia makan dengan metodh dari *parentclass*nya (manusia).

Akan tetapi ketika dia kerja, dia kerja dengan metodh baru yang didefinisikan khusus pada kelas programmer ("Tak…Tak…Klik.. " bukan "Kerja….kerjaaa…"), inilah yang disebut dengan *metodh overriding.*

Lainhalnya dengan Andi, Badu adalah seorang tentara berpangkat kopral keturunan manusia. Suatu saat dia ditanya atasannya nama dan pangkatnya, kemudian dia disuruh makan dan kemudian diperintah untuk kerja, maka kode javanya :

```
/*
Disimpan dalam file "badu.java"
*/

public class badu
{
      public static void main(String arg[])
      {
      tentara badu= new tentara("Badu","Kopral");
      System.out.println("Nama= "+ badu.tampilkanNama());
      System.out.println("Pangkat= "+ badu.tampilkanPangkat());
      badu.makan();
      badu.kerja();
      }
}
```

Dalam kasus badu, selain badu mewarisi atribut nama dari manusia, sebagai tentara dia mempunyai atribut pangkat. Jadi dalam pewarisan, kita bisa menambah atribut-atribut baru dan juga bisa menambahkan metodh-metodh baru, bahkan meng*override* metodh yang ada pada *parentclassnya.*

Metodh Final

Metodh final adalah metodh yang tidak bisa dioverride oleh *subclass*nya. Pendefinisian metodh final dengan cara menambahkan keyword final didepan definisi metodh tersebut, seperti pada contoh berikut

```
Public final void tidur();
 System.out.println("Zzz..Zzzz..Zzzzzzz");
```

Dengan cara seperti ini maka kita mendefinisikan bahwa metodh tidur adalah dengan cara

```
      System.out.println("Zzz..Zzzz..Zzzzzzz");
```

Baik dilakukan oleh kelasnya sendiri maupun oleh anak cucu kelasnya.

C. Praktikum

1. Buatlah turunan dari kelas mahasiswa yang telah Anda buat pada modul 1 menjadi kelas: Pecinta alam dan buatlah atribut dan metodh-metodhnya masing-masing sebanyak 2 buah
2. Buatlah 2 objek untuk menginstantiasi kelas pecinta alam dengan atribut nama dan npm Anda sendiri dan 1 teman Anda dengan mengimplementasikan beberapa metodh yang telah Anda definisikan dalam kelas kelasnya.

Kompile file kelas pecinta alam dan objek-objek yang telah Anda buat, kemudian jalankan objek-objek yang telah Anda buat !

Modul 3. Polymorfisme

A. Tujuan

Praktikan diharapkan dapat:

1. Memahami dan mengimplementasikan polymorfisme

B. Teori Singkat

Salah satu pilar Pemrograman Berorientasi Objek adalah polymorfisme yaitu kemampuan beberapa objek bertipe sama bereaksi secara berbeda terhadap "pesan" yang sama.

Sebagai contoh kita tambah lagi turunan dari manusia yaitu kelas sopir. Kelas sopir diimplementasikan dalam java

```
/*
Disimpan dalam file "sopir.java"
*/

public class sopir extends manusia
{
      public sopir(String n)
      {
            super(n);
      }

      public void kerja()
      {
      System.out.println("Ngung... Ngung... Ngung...Ciiit..");
      }
}
```

Dedi adalah seorang sopir keturunan manusia, untuk menginstantiasi objek dedi ditunjukkan dalam kode berikut.

```
/*
Disimpan dalam file "dedi.java"
*/

public class dedi
{
      public static void main(String arg[])
      {
      sopir dedi= new sopir("Dedi");
      System.out.println("Nama= "+ dedi.tampilkanNama());
      dedi.makan();
      dedi.kerja();
      }
}
```

Kemudian Andi sang programmer, Badu sang tentara dan Dedi sang sopir diperintahkan untuk bekerja, apa reaksinya?

Untuk melihat reaksi masing-masing, perhatikan kode java berikut!

```
/*
Disimpan dalam file "pekerja.java"
*/

public class pekerja
{
      public static void main(String args[])
      {
      manusia[] profesi= new manusia[3];
      profesi[0]=new programmer("Andii");
      profesi[1]=new tentara("Badu","Kopral");
      profesi[2]=new sopir("Dedi");

      for (int i=0; i<3; i++)
      {
            profesi[i].kerja();
      }
      }
}
```

Hasil eksekusi para pekerja adalah sebagai berikut:

```
Tak…Tak…Klik...
Dor... Dor... Dor…
Ngung... Ngung... Ngung...Ciiit..
```

Message sama yang dikirimkan ke objek berbeda akan menghasilkan hasil yang berbeda, inilah yang disebut ***polymorfisme***.

C. Praktikum

1. Buatlah kelas Rohis yang merupakan kelas turunan dari mahasiswa dan buatlah metodh-metodhnya.
2. Buatlah 1 objek untuk menginstantiasi kelas rohis dan mengimplementasikan beberapa metodh yang telah Anda definisikan dalam kelas rohis.
3. Panggilllah objek yang telah Anda buat pada kelas pecinta alam dan rohis untuk bekerja sehingga tampil efek-efek *polymorfisme*

Modul 4. Pengkapsulan

A. Tujuan

Praktikan diharapkan dapat:
1. Memahami dan menggunakan dengan benar hak akses public, protected dan private
2. Memahami kegunaan pengaturan hak akses dalam pengembangan aplikasi

B. Teori Singkat

Pilar terakhir dari 3 pilar Pemrograman Berorientasi Objek adalah pengkapsulan, dimana pengembang software dapat menyembunyikan detail suatu objek.

Hak akses public memungkinkan semua kelas mengaksesnya, hak akses protected hanya diberikan kepada kelasnya sendiri dan turunannya, serta kelas-kelas dalam satu paket. sedangkan private hanya boleh diakses oleh kelasnya sendiri.

Perhatikan kelas manusia dibawah ini!

```
/*
Disimpan dalam file "manusia.java"
*/

class manusia
{
      public String nama;

      public manusia(String n)
      {
      this.nama         = n;
      }

      public String tampilkanNama()
      {
      return nama;
      }

      public void makan()
      {
      System.out.println("Nyam... nyam... nyam...");
      }

      public void kerja()
      {
      System.out.println("Kerja... kerjaaa...");
      }

      private  void bunuhDiri()
      {
      System.out.println("Dor…bruk...");
      }

}
```

Andi adalah objek bentukan dari kelas manusia

```
/*
Disimpan dalam file "andi.java"
*/

class andi
{

      public static void main(String arg[])
      {
      manusia andi= new manusia("Andi");
      System.out.println("Nama= "+ andi.tampilkanNama());
      andi.makan();
      }
}
```

Apa yang terjadi jika hak akses private diakses oleh kelas lain?

C. Praktikum

1. Kompile dan jalankan kelas manusia dan andi!
2. Ubah hak akses makan dari public menjadi protected dan ulangi praktikum nomor 1! Apa yang terjadi? **Jelaskan!**
3. Tambahkan pada kelas andi untuk memanggil metodh bunuh diri, ulangi praktikum no 1. Apa yang terjadi? **Jelaskan!**

Modul 5. Interface

A. Tujuan

Praktikan diharapkan dapat:
1. Memahami konsep pewarisan ganda
2. Memahami konsep interface
3. Mengimplementasikan Interface pada Java untuk melakukan pewarisan ganda

C. Teori Singkat

Untuk membuat suatu kelas dapat kita turunkan dengan pewarisan field-field dan metodh pada base classnya. Bagaimana kita membuat kelas yang menurunkan sifat dari beberapa base class? misalkan kita akan membuat kelas superman yang dia bisa membuat program layaknya programmer, dia juga ahli menggunakan senjata layaknya tentara, bahkan dia bisa terbang seperti elang (keturunan binatang)? Caranya adalah dengan pewarisan ganda. Dalam Java tidak dikenal pewarisan ganda, sehingga digunakan interface.

Contoh pewarisan ganda yang tidak benar dalam Java

```
class superman extends programmer, tentara, burung
 {
 }
```

Perhatikan contoh berikut:

```
/*
Disimpan dalam file "superman.java"
*/

interface programmer {
  void memrogram();
}

interface tentara {
  void menembak();
}

interface burung {
  void terbang();
  void buangKotoran();

}

class superman implements programmer,tentara,burung {

public void memrogram(){};
public void menembak(){};
public void terbang(){};
public void buangKotoran(){};
}
```

Kemudian kelas superman diinstantiasi menjadi objek bernama bejo, contoh kode program javanya sebagai berikut:

```
/*
Disimpan dalam file "bejo.java"
*/

public class bejo
{
      public static void main(String arg[])
      {
      superman bj= new superman();

      bj.memrogram();
      {
      System.out.println("Implementasi memrogram ...tak..tik");
      }

      bj.menembak();
      {
      System.out.println("Implementasi menembak ...dor..dor");
      }

      bj.terbang();
      {
      System.out.println("Implementasi terbang......Zap....");
      }
      }
}
```

Dalam kode diatas bejo menentukan sendiri cara mengimplementasi beberapa metodh dari interface yang telah didefiniskan dalam kelas superman (misalkan cara memrogramnya bagaimana, cara menembaknya bagaimana dan cara terbangnya seperti apa), selain itu bejo tidak berminat untuk mengimplementasikan metodh buangKotorang dari interface burung (misalnya karena burung biasa membuang kotorang di sembarang tempat).

Jadi interface dapat dianalogikan seperti menandatangani kontrak kerja, misalnya sebagai dosen dia wajib mengajar, membuat soal ujian dsb, akan tetapi cara mengajarnya dan membuat soalnya dilakukan terserah masing-masing dosen (tidak ditentukan dalam kontrak kerja)

C. Praktikum

1. Buatlah kelas mahasiswaSuper dengan menggunakan interface mahasiswa, atlit, wiraswasta, dan kemudian instantiasi menjadi objek dengan nama sesuai nama Anda

Modul 6. Pemaketan

A. Tujuan

Praktikan diharapkan dapat:
1. Memahami dan mengimplementasikan pemaketan
2. Memahami dan menggunakan kelas yang terdapat dalam paket tertentu

B. Teori Singkat

Ketika kelas-kelas yang dibuat semakin banyak dan semakin banyak, hal ini akan membuat struktur program menjadi rumit kalau tidak dikelola dengan baik. Untuk itu kelas-kelas disimpan dalam paket-paket tertentu, misalkan kelas programmerC, programmerJava, programmerPHP berada/disimpan dalam paket programmer. Sementara marinir, kopasus, paskhas berada dalam paket tentara. Keuntungan pengaturan program dalam nama paket-paket adalah:
1. Terhindar dari konflik penamaan. Misalkan saja dalam membuat program kita menggunakan/mengimport kelas dari luar yang dibuat oleh programmer lain, sehingga mungkin saja dalam penamaan kelas terjadi persamaan. Dengan menunjukkan nama lengkap paket/kelasnya maka tidak akan terjadi konflik penamaan
2. Teratur. Misalkan paket dosen terdapat kelas dosenPBO, dosenKalkulus, dosenEtika. Paket tentara terdapat marinir, kopasus, paskhas. Dengan struktur demikian, maka akan mempermudah ketika ketika kita akan menggunakan/mengimport kelas, misalkan saja kita akan mengimport kelas kopasus, tentunya kita mencari dalam paket tentara-bukan paket dosen.

Sebagai contoh kita akan membuat 2 paket: paket programmer dan paket tentara. Paket programmer terdiri dari kelas programmerC dan programmerJava.

```
/*
Disimpan dalam "programmer/programmerC.java"
*/

package programmer;

public class programmerC
{
      public programmerC()
      {
      }

      public void kerja()
      {
      System.out.println("Implementasi metodh kerja Programmer C ..");
      }

      public void bersantai()
      {
      System.out.println("Implementasi metodh bersantai Programmer .. ");
      }
}


/*
Disimpan dalam file "programmer/programmerJava.java"
*/

package programmer;
```

```
public class programmerJava
{
      public programmerJava()
      {
      }

      public void kerja()
      {
      System.out.println("Implementasi metodh kerja Programmer Java ..");
      }

      public void bersantai()
      {
      System.out.println("Implementasi metodh bersantai Programmer Java.. ");
      }
}
```

Sedangkan paket tentara terdapat kelas AngkatanDarat dan AngkatanLaut.

```
/*
Disimpan dalam file "tentara/kopasus.java"
*/

package tentara;

public class kopasus
{
      public kopasus()
      {
      }

      public void kerja()
      {
      System.out.println("Implementasi metodh kerja kopasus ....");
      }

      public void bersantai()
      {
      System.out.println("Implementasi metodh bersantai kopasus");
      }
}
```

Kalau dieksekusi *class* Andi akan menghasilkan

```
/*
Disimpan dalam file "tentara/marinir.java"
*/

package tentara;

public class marinir
{
      public marinir()
      {
      }

      public void kerja()
      {
```

```
      System.out.println("Implementasi metodh kerja marinir ..");
      }

      public void bersantai()
      {
      System.out.println("Implementasi metodh bersantai marinir ..");
      }
}
```

Mengimport kelas
Untuk dapat mengimport kelas digunakan keyword import [nama paketnya]. Sebagai contoh instantiasi kelas programmerJava dalam paket programmer menjadi objek ahmed.

```
/*
Disimpan dalam file "ahmed.java"
*/

import programmer.programmerC;

class ahmed
{
      public static void main(String arg[])
      {
      programmerC ahmed= new programmerC();
      ahmed.kerja();
      }
}
```

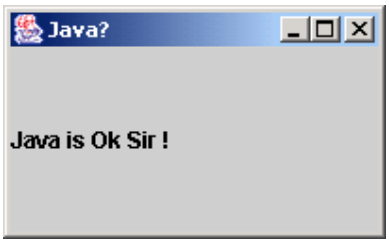
sehingga hasil eksekusinya adalah;

```
Implementasi metodh kerja Programmer C ....
```

Banyak sekali kelas-kelas yang telah dibuat oleh **Sun** yang dapat kita gunakan, misalkan untuk membuat windows bisa digunakan/import paket awt dan swing.

```
/*
Disimpan dalam file "JavaOk.java"
*/

import javax.swing.*;

public class JavaOk {

public static void main(String[] args) {
        JFrame frame = new JFrame("Java?");
        final JLabel label = new JLabel("Java is Ok Sir");
        frame.getContentPane().add(label);

        frame.pack();
        frame.setVisible(true);
    }
}
```

yang akan menampilkan hasi sebagai berikut:



C. Praktikum

1. Buatlah paket mahasiswa yang isinya kelas IF, kelas MI, kelas SI dan buat metodh-metodhnya
2. Buatlah paket orangTua Siswa yang isinya kelas petani, nelayan, wiraswasta atau yang lainnya dan buat metodhnya.
3. Instantiasi sebuah objek dan beri nama dengan nama Anda yang mengimport dari paket mahasiswa
4. Instantiasi sebuah objek dan beri nama dengan nama Orang tua Anda yang mengimport dari paket orangTuaSiswa

Lampiran.
Konfigurasi PATH dan Penggunaan Java

Konfigurasi PATH dan CLASSPATH

Setelah Java Development Kit terinstall (misalkan di C:\JDK), konfigurasilah file **`autoexec.bat`** dengan menyeting PATH dan CLASSPATH nya.

```
SET PATH=%PATH%;C:\JDK\bin
SET CLASSPATH=.;"C:\JDK\lib"
```

Pada setting PATH dimaksudkan agar file-file executable di direktori ”C:\JDK\bin” bisa dijalankan dari seluruh direktori kerja.

Pada setting CLASSPATH dimaksudkan agar class-class yang ada di “C:\JDK\lib” bisa diimport dari seluruh direktori kerja.

Perintah-Perintah Dasar

1. **Mengkompile file java**

```
Prompt:\javac [option] [source file]
```

Option boleh ada dan boleh juga tidak

Contoh: Untuk mengkompile file “tes.java”

```
Prompt:\javac tes.java
```

2. **Menjalankan class (hasil kompilasi)**

```
Prompt:\Java [-option] nama class
```

Option boleh ada dan boleh juga tidak

Contoh untuk menjalankan tes.class

```
Prompt:\Java tes
```

Baca manual/dokumentasi programnya untuk keterangan selengkapnya!